```
public void sort(int[] nums){
      for (int i=0; i<nums.length; i++){                              //line A
         for (int j=0; j<nums.length - i - 1; j++){                   //line B
            if (nums[j] > nums[j+1]){                                 //line C
               int temp = nums[j];                                    //line D
               nums[j] = nums[j+1];
               nums[j+1] = temp;
            }
         }//end j loop
      }//end i loop
   }
```

```
public void sort(int[] nums){
     for (int i=0; i<nums.length-1; i++){
        int posOfLowest = i;                                   //line A
        for (int j=i+1; j<nums.length; j++){
           if (nums[j] < nums[posOfLowest])                    //line B
              posOfLowest = j;
        }
        int temp = nums[i];                                    //line C
        nums[i] = nums[posOfLowest];
        nums[posOfLowest] = temp;
     }
}
```

```
public void sort(int[] nums){
   for (int i = 1; i < nums.length; i++){
       int j = i;                                          //line A
       int B = nums[i];
       while ( (j > 0) && (nums[j-1] > B) ){
              nums[j] = nums[j-1];                         //line B
              j--;
       }
       nums[j] = B;                                        //line C
   }
}
```

```
public int search(int[] A, int x) {
      for(int k=0; k<A.length; k++)
         if (A[k]==x)
            return(k);
      return(-1);
}
```

more on next page...

```
public int search(int[] A, int x) {
        int lo = 0;
        int hi = A.length - 1;
        while (lo <= hi) {
            int mid = lo + (hi - lo) / 2;           //line A
            if (x < A[mid])
                hi = mid - 1;                        //line B
            else if (x > A[mid])
                lo = mid + 1;                        //line C
            else
               return mid;
        }
        return(-1);
    }
```